CNN NETWORKS FOR STATE ESTIMATION:

APPLICATION TO THE ESTIMATION OF PROJECTILE TRAJECTORY WITH AN INVARIANT EXTENDED KALMAN FILTER

Workshop conjoint GdR-MACS du CNRS & COMET-SCA du CNES

10 et 11 juin 2021

Alicia ROUX

alicia.roux@uha.fr

Supervised by :
Dr. Sébastien CHANGEY, ISL
Dr. Jonathan WEBER, UHA
Pr. Jean-Philippe LAUFFENBURGER, UHA

CONTENTS

I. INTRODUCTION

1. Statement
2. State of art
3. Algorithm

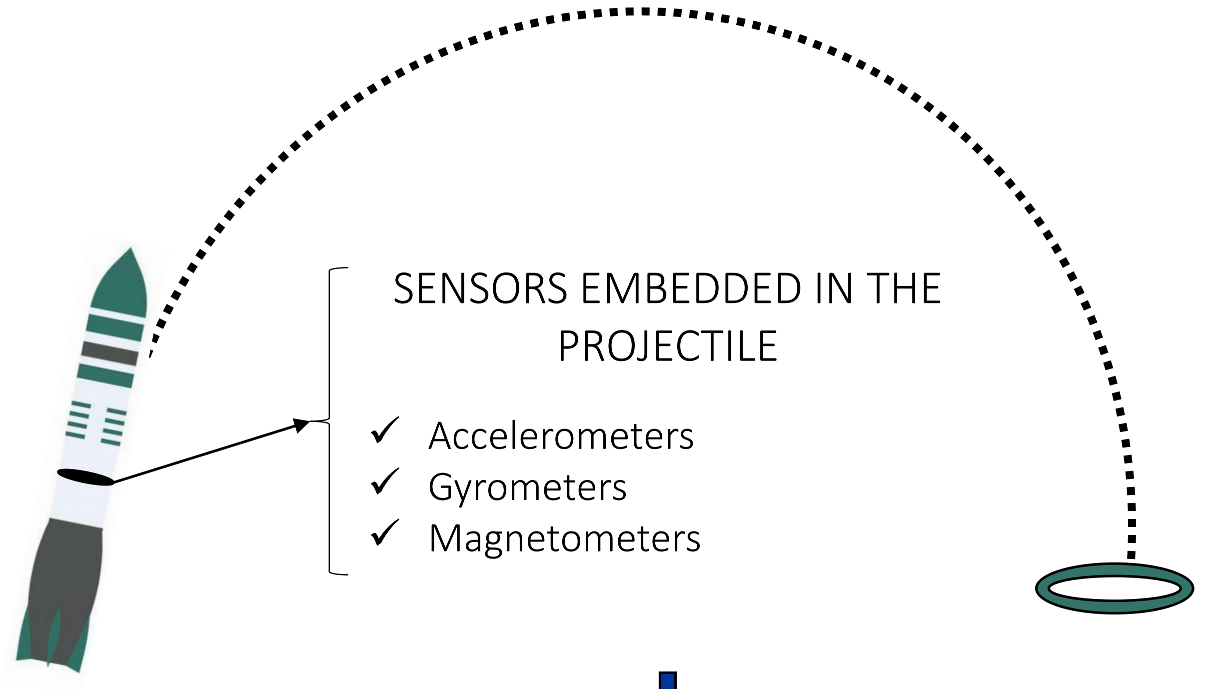
STATEMENT

THESIS TOPIC

STUDY OF ARTIFICIAL INTELLIGENCE METHODS FOR FLYING OBJECTS NAVIGATION

Since October 2020

Supervised by :
- Pr. Jean-Philippe LAUFFENBURGER, UHA
- Dr. Jonathan WEBER, UHA
- Dr. Sébastien CHANGEY, ISL



AUTOMATIC CONTROL + **ARTIFICIAL INTELLIGNCE**

ACCURATE AND LOW COST NAVIGATION SOLUTION

STATE OF THE ART
NAVIGATION SOLUTIONS

PROJECTILE

➡ GNSS/INS INTEGRATION

GROUND VEHICLE: inertial navigation solution

➡ DEAD RECKONING

Discrepancy in estimates due to accumulations of sensor errors

➡ KALMAN FILTER

EXTENDED KALMAN FILTERS

Filter convergence not guaranteed

STATE OF THE ART
NAVIGATION SOLUTIONS

GNSS DENIED

PROJECTILE

➡ GNSS/INS INTEGRATION

GROUND VEHICLE: inertial navigation solution

➡ DEAD RECKONING

Discrepancy in estimates due to accumulations of sensor errors

➡ KALMAN FILTER

EXTENDED KALMAN FILTERS

Filter convergence not guaranteed

INERTIAL NAVIGATION SOLUTION

INVARIANT EXTENDED KALMAN FILTER
IMPERFECT R-IEKF

↓

Invariant observer defined on a Lie group

↓

Invariant estimation error

↓

Convergent nonlinear observer

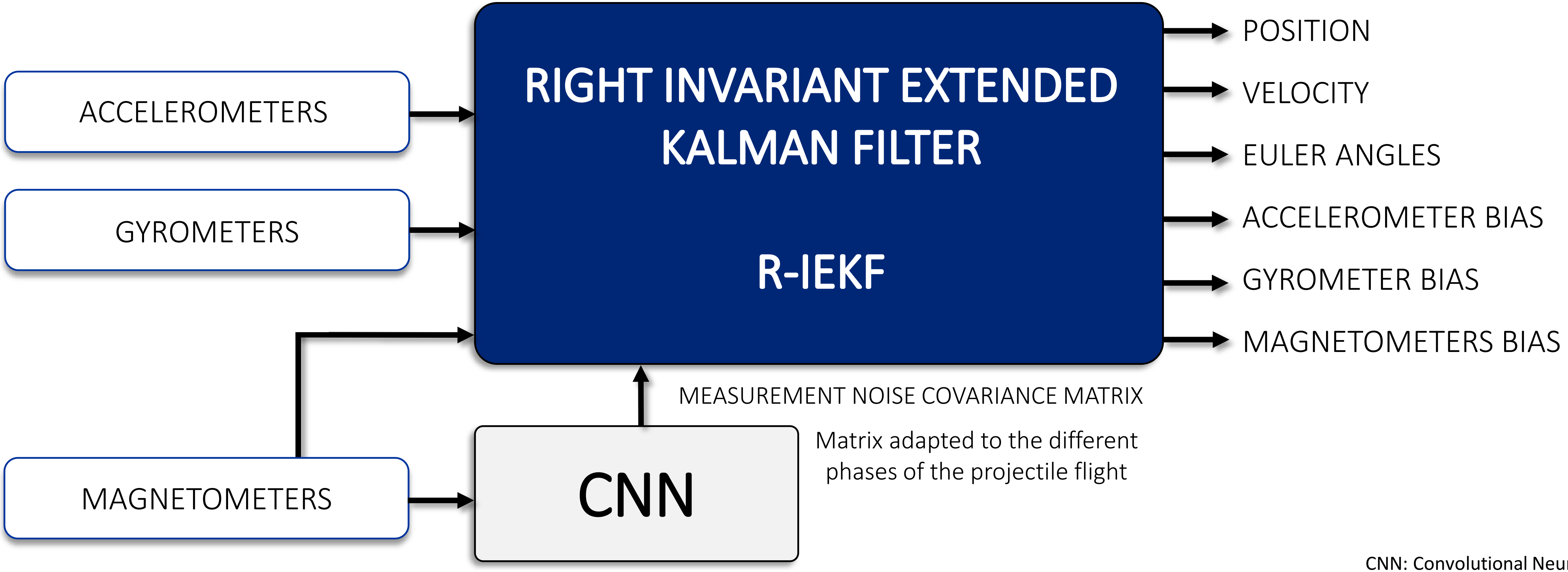
[1] Paul D Groves. Principles of gnss, inertial, and multisensor integrated navigation systems, 2015.
[2] Axel Barrau. Non-linear state error based extended Kalman filters with applications to navigation, 2015.

ALGORITHM

ALGORITHM FOR ESTIMATING THE PROJECTILE TRAJECTORY IN THE NAVIGATION FRAME

ALGORITHM

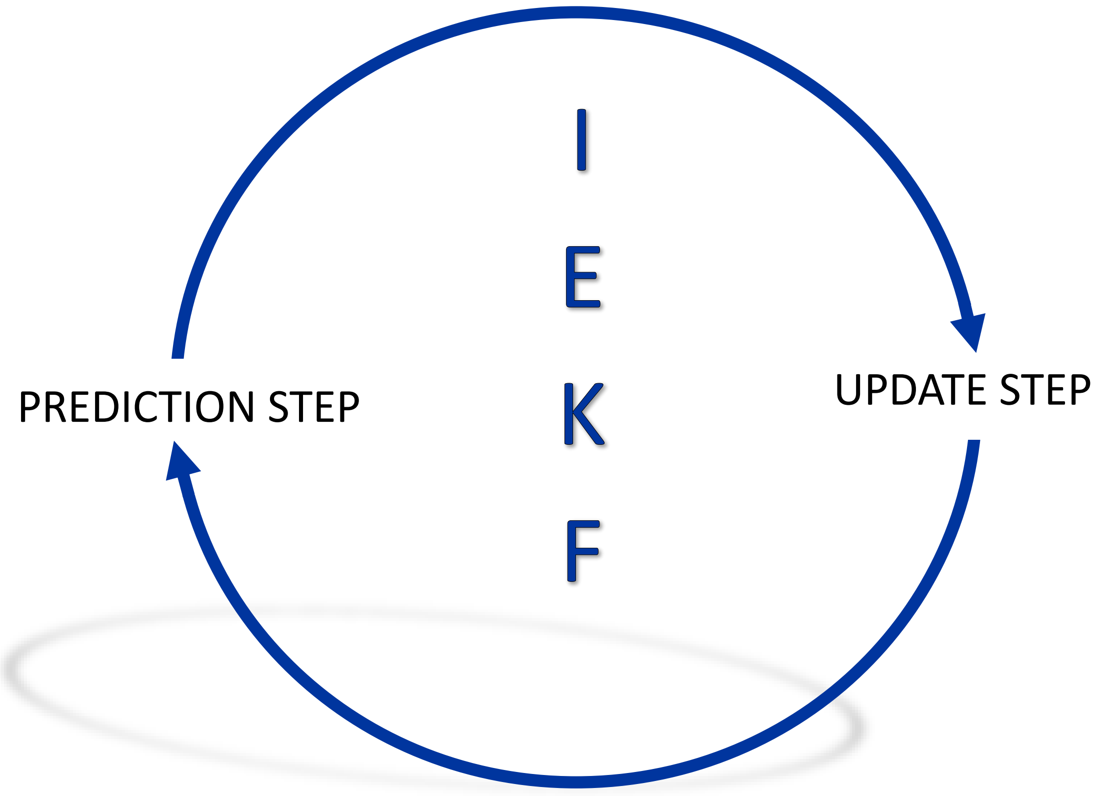
ALGORITHM FOR ESTIMATING THE PROJECTILE TRAJECTORY IN THE NAVIGATION FRAME



CNN: Convolutional Neural Networks

II. INVARIANT EXTENDED KALMAN FILTER: IEKF

INTRODUCTION



[3] Axel Barrau, Silvère Bonnabel. The invariant extended kalman filter as a stable observer, 2016.

INTRODUCTION

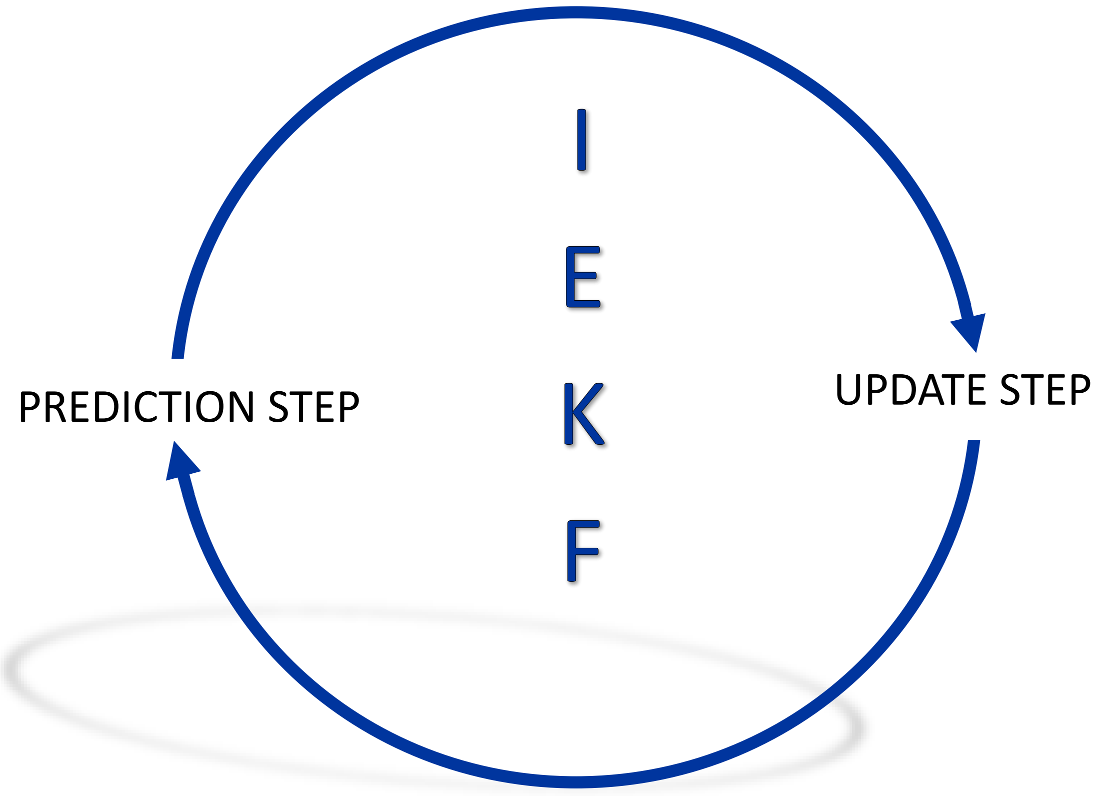
SYSTEM

Let $G \subset \mathbb{R}^{n.n}$ be a Lie group whose associated algebra is denoted by $\mathfrak{g}$.

Let the system be defined on a Lie group G such as:

$$\frac{d}{dt}\mathcal{X}_t = f_{u_t}(\mathcal{X}_t) + \mathcal{X}_t w_t$$

With $\mathcal{X}_t \in G, w_t \in \mathfrak{g}$.



[3] Axel Barrau, Silvère Bonnabel. The invariant extended kalman filter as a stable observer, 2016.

INTRODUCTION

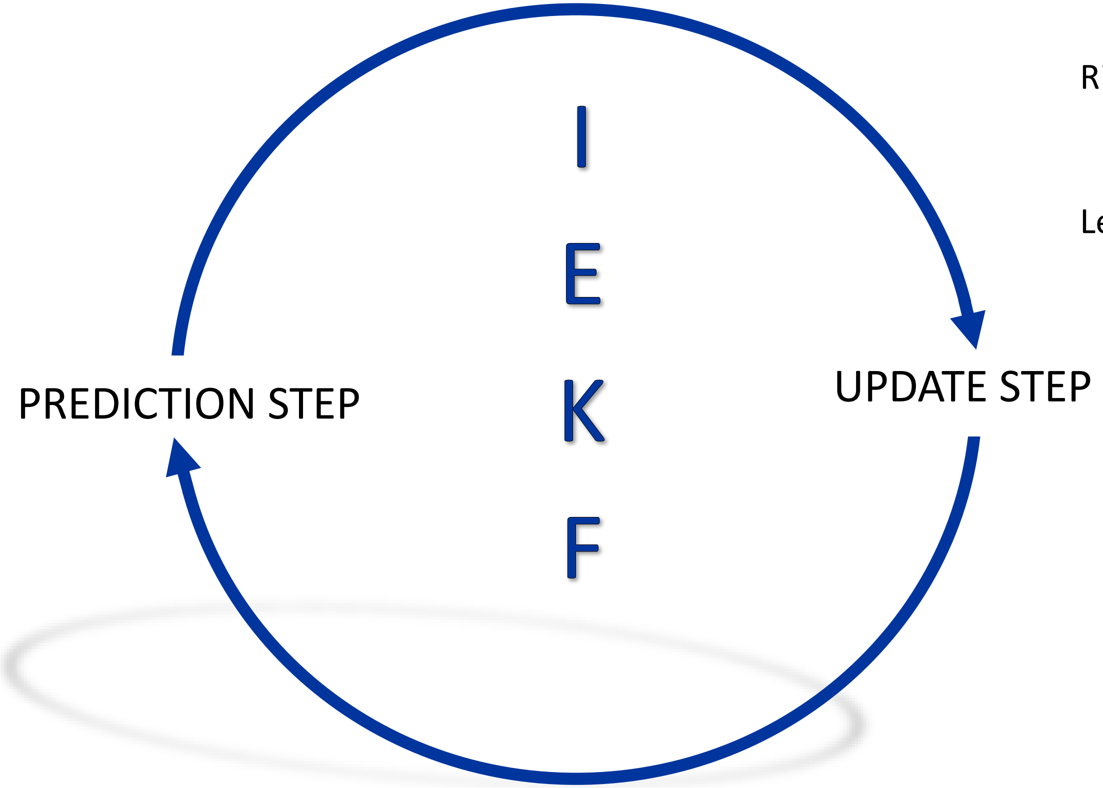
SYSTEM

Let $G \subset \mathbb{R}^{n.n}$ be a Lie group whose associated algebra is denoted by $\mathfrak{g}$.

Let the system be defined on a Lie group G such as:

$$\frac{d}{dt}\mathcal{X}_t = f_{u_t}(\mathcal{X}_t) + \mathcal{X}_t w_t$$

With $\mathcal{X}_t \in G, w_t \in \mathfrak{g}$.



OBSERVATIONS

Right invariant observation:

$$Y_{t_n}{}^{k} = \mathcal{X}_t{}^{-1}(d^k + V_n^k) + B_n^k$$

Left invariant observation:

$$Y_{t_n}{}^{k} = \mathcal{X}_t(d^k + B_n^k) + V_n^k$$

R-IEKF: Right-Invariant Extended Kalman Filter
L-IEKF: Left-Invariant Extended Kalman Filter

[3] Axel Barrau, Silvère Bonnabel. The invariant extended kalman filter as a stable observer, 2016.

INTRODUCTION

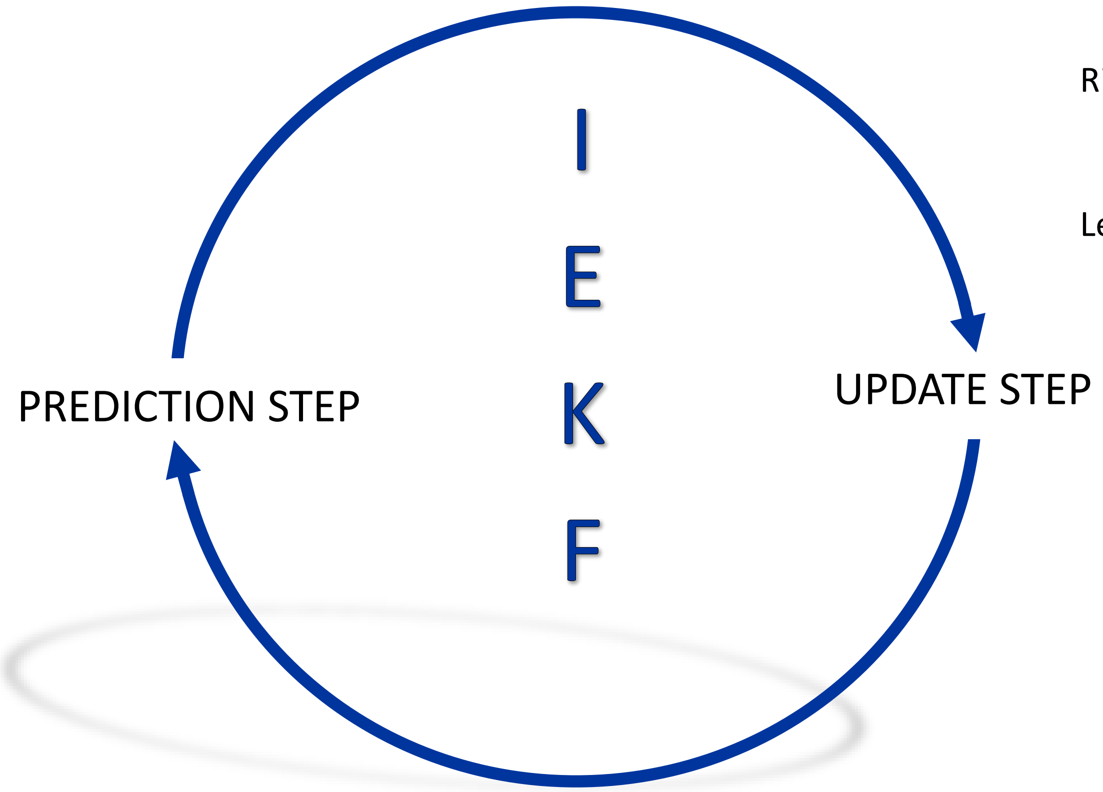
SYSTEM

Let $G \subset \mathbb{R}^{n.n}$ be a Lie group whose associated algebra is denoted by $\mathfrak{g}$.

Let the system be defined on a Lie group G such as:

$$\frac{d}{dt}\mathcal{X}_t = f_{u_t}(\mathcal{X}_t) + \mathcal{X}_t w_t$$

With $\mathcal{X}_t \in G, w_t \in \mathfrak{g}$.



OBSERVATIONS

Right invariant observation:

$$Y_{t_n}{}^{k} = \mathcal{X}_t{}^{-1}(d^k + V_n^k) + B_n^k$$

Left invariant observation:

$$Y_{t_n}{}^{k} = \mathcal{X}_t(d^k + B_n^k) + V_n^k$$

INVARIANT ERROR

Right invariant error:

$$\eta_t{}^R = \hat{\mathcal{X}}_t \mathcal{X}_t{}^{-1}$$

Left invariant error:

$$\eta_t{}^L = \mathcal{X}_t{}^{-1} \hat{\mathcal{X}}_t$$

R-IEKF: Right-Invariant Extended Kalman Filter
L-IEKF: Left-Invariant Extended Kalman Filter

[3] Axel Barrau, Silvère Bonnabel. The invariant extended kalman filter as a stable observer, 2016.

INTRODUCTION

PREDICTION STEP

Prediction step similar to an EKF

System is group affine, then the error dynamics are independent of state

SYSTEM

Let the system be defined on a Lie group G such as:

$$\frac{d}{dt}\mathcal{X}_t = f_{u_t}(\mathcal{X}_t) + \mathcal{X}_t w_t$$

With $\mathcal{X}_t \in G, w_t \in \mathfrak{g}$.

STATE PREDICTION

$$\frac{d}{dt}\hat{\mathcal{X}}_t = f_{u_t}(\hat{\mathcal{X}}_t)$$

COVARIANCE MATRIX PREDICTION

$$\frac{d}{dt}P_t = A_t P_t + P_t {A_t}^T + \hat{Q}_t$$

$\hat{Q}_t$ covariance matrix of the modified process noise

(log-linear property) : $\eta_t = exp(\xi_t)$

$$\frac{d}{dt}\xi_t = A_t \xi_t + \hat{w}_t$$

[3] Axel Barrau, Silvère Bonnabel. The invariant extended kalman filter as a stable observer, 2016.

INTRODUCTION

UPDATE STEP

KALMAN MATRIX

$$K_k = P_{k|k-1} {H_k}^T \left(H_k P_{k|k-1} {H_k}^T + \hat{N}_k \right)^{-1}$$

Observation matrix H_k determined from observations and linearized error ξ_t

$\hat{N}_k$ covariance matrix of the modified measurement noise

UPDATE ESTIMATES

L-IEKF

$$\hat{\mathcal{X}}_{k|k} = \hat{\mathcal{X}}_{k|k-1} exp \left(K_k \begin{pmatrix} {\hat{\mathcal{X}}_{k|k-1}}^{-1} {Y_{t_k}}^{1} - d^1 \\ \dots \\ {\hat{\mathcal{X}}_{k|k-1}}^{-1} {Y_{t_k}}^{n} - d^n \end{pmatrix} \right)$$

R-IEKF

$$\hat{\mathcal{X}}_{k|k} = exp \left(K_k \begin{pmatrix} \hat{\mathcal{X}}_{k|k-1} {Y_{t_k}}^{1} - d^1 \\ \dots \\ \hat{\mathcal{X}}_{k|k-1} {Y_{t_k}}^{n} - d^n \end{pmatrix} \right) \hat{\mathcal{X}}_{k|k-1}$$

COVARIANCE MATRIX UPDATE

$$P_{k|k} = (\mathbb{I} - K_k H_k) P_{k|k-1}$$

[3] Axel Barrau, Silvère Bonnabel. The invariant extended kalman filter as a stable observer, 2016.

RIGHT-INVARIANT EXTENDED KALMAN FILTER
WITH IMU MEASUREMENTS ONLY

IMU

- NOISY GYROMETERS → $\widetilde{\omega}_b$
- NOISY ACCELEROMETERS → $\tilde{a}_b$
- NOISY MAGNETOMETERS → $H_n = RH_b + b_H$

R-IEKF

INERTIAL NAVIGATION EQUATIONS

$$\dot{R} = R[\widetilde{\omega}_b - b_\omega - W_\omega]_\times$$

$$\dot{v} = R(\tilde{a}_b - b_a - W_a) + g$$

$$\dot{p} = v$$

$$\dot{b}_\omega = W_{b_\omega}$$

$$\dot{b}_a = W_{b_a}$$

$$\dot{b}_H = W_{b_H}$$

Outputs:

- Projectile position $p_{k|k}$
- Projectile velocity $v_{k|k}$
- Projectile Euler angles (roll $\phi_{k|k}$, pitch $\theta_{k|k}$, yaw $\psi_{k|k}$)
- Accelerometer bias $b_{a_{k|k}}$
- Gyrometer bias $b_{\omega_{k|k}}$
- Magnetometers bias $b_{H_{k|k}}$

IMU: Inertial Measurement Unit

III. CONVOLUTIONAL NEURAL NETWORKS

1. Generalities
2. Network structure
3. Dataset

GENERALITIES

GOAL: Estimate the measurement noise covariance matrix

- Matrix variable over time
- Adapted to the different phases of the projectile flight

[4] Martin Brossard. Deep learning, Inertial Measurements Units, and Odometry : Some Modern Prototyping Techniques for Navigation Based on Multi-Sensor Fusion, 2020.

GENERALITIES

GOAL: Estimate the measurement noise covariance matrix

- Matrix variable over time
- Adapted to the different phases of the projectile flight

MAGNETOMETER READINGS
IN THE PROJECTILE FRAME

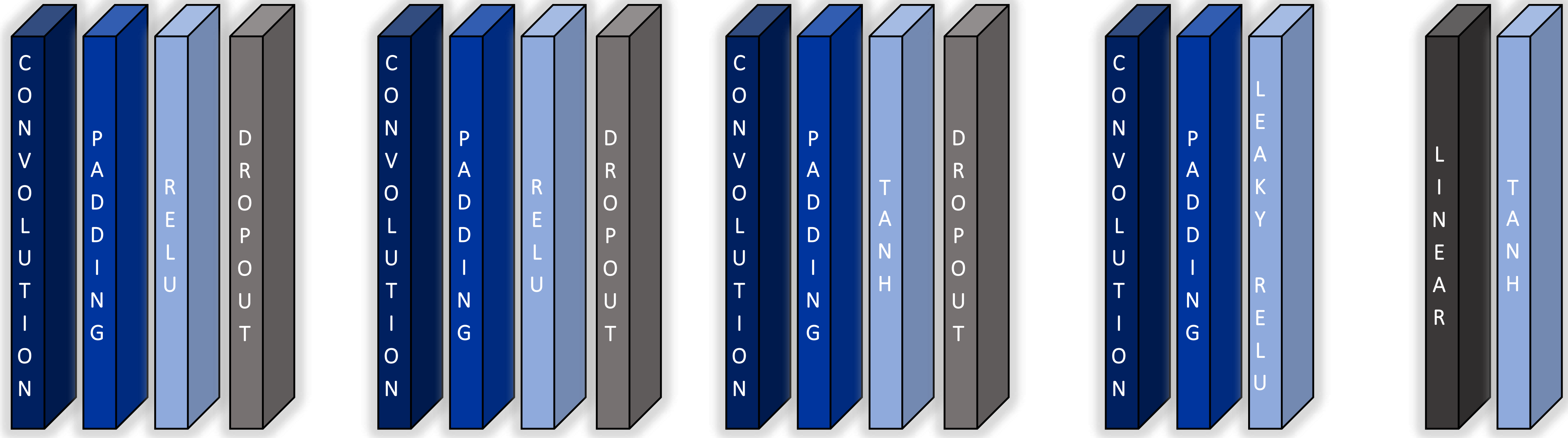
MAGNETOMETERS H_{b_k} → CNN → $\begin{bmatrix} \alpha_k \\ \beta_k \\ \gamma_k \end{bmatrix}$ →

MEASUREMENT NOISE COVARIANCE MATRIX

$$\begin{bmatrix} \sigma_{H_x}^{\alpha_k} & 0 & 0 \\ 0 & \sigma_{H_x}^{\beta_k} & 0 \\ 0 & 0 & \sigma_{H_x}^{\gamma_k} \end{bmatrix}$$

[4] Martin Brossard. Deep learning, Inertial Measurements Units, and Odometry : Some Modern Prototyping Techniques for Navigation Based on Multi-Sensor Fusion, 2020.

NETWORK STRUCTURE



OPTIMIZATION ALGORITHM
ADAM

LEARNING RATE
10^{-3}

EPOCH
10

LOSS FUNCTION
MEAN SQUARED ERROR
$l_n = (x_n - y_n)^2$

DATASET

ISL DATASET (BALCO)

- 8000 mortar fire simulations
- One simulation ≈ 36000 datas

Measurement	Frame	Additive noise
Accelerometers $\tilde{a}_b$	Projectile frame ***b***	Gaussian white noise
Gyrometers $\tilde{\omega}_b$	Projectile frame ***b***	Gaussian white noise
Magnetometers H_b	Projectile frame ***b***	Gaussian white noise
Reference magnetic field H_n	Navigation frame ***n***	
Position P_n	Navigation frame ***n***	
Velocity V_n	Navigation frame ***n***	
Euler angles : roll ϕ, pitch θ, yaw ψ	Navigation frame ***n***	

DATASET

ISL DATASET (BALCO)

- 8000 mortar fire simulations
- One simulation ≈ 36000 datas

Measurement	Frame	Additive noise
Accelerometers $\tilde{a}_b$	Projectile frame ***b***	Gaussian white noise
Gyrometers $\tilde{\omega}_b$	Projectile frame ***b***	Gaussian white noise
Magnetometers H_b	Projectile frame ***b***	Gaussian white noise
Reference magnetic field H_n	Navigation frame ***n***	
Position P_n	Navigation frame ***n***	
Velocity V_n	Navigation frame ***n***	
Euler angles : roll ϕ, pitch θ, yaw ψ	Navigation frame ***n***	

TRAINING DATASET

200 mortar fire simulations

TEST DATASET

20 mortar fire simulations

COMPUTER SPECIFICATIONS

- Operating System (Linux Ubuntu 18.04.4 LTS)
- Memory (31 GB)
- Processor (Intel®Xeon (R))
- Graphics card (NVIDIA QUADRO RTX 5000).

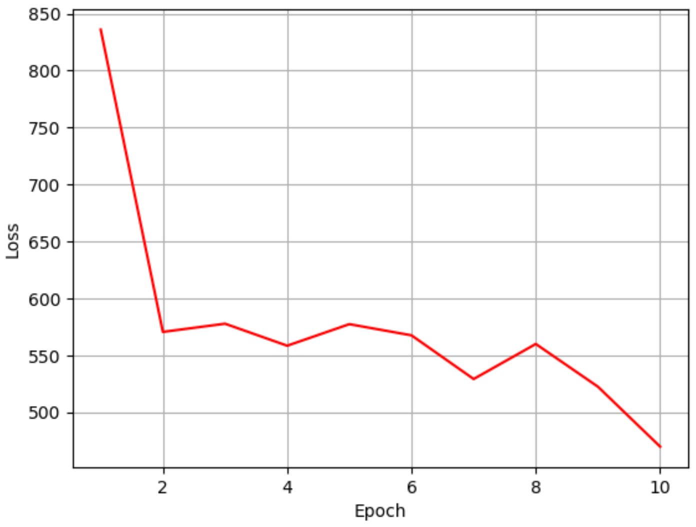
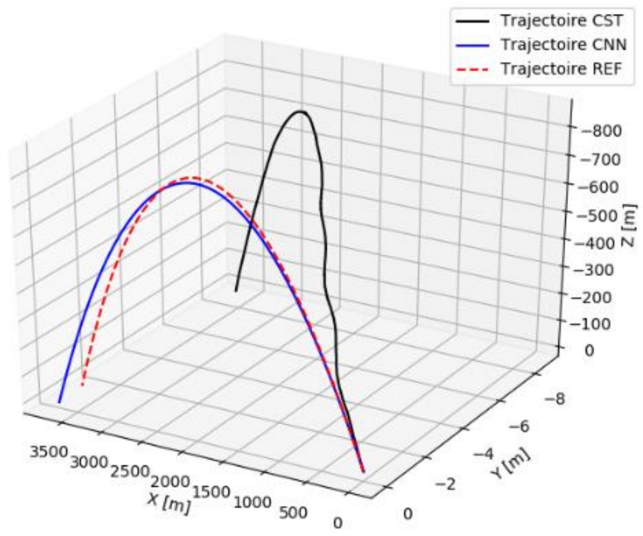


Fig1. Loss evolution during training



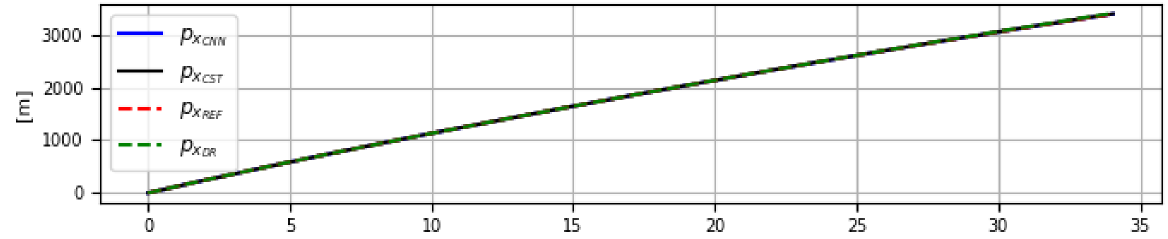
IV. VALIDATION

1. Estimation results
2. Test dataset: analysis
3. Test dataset: error precision

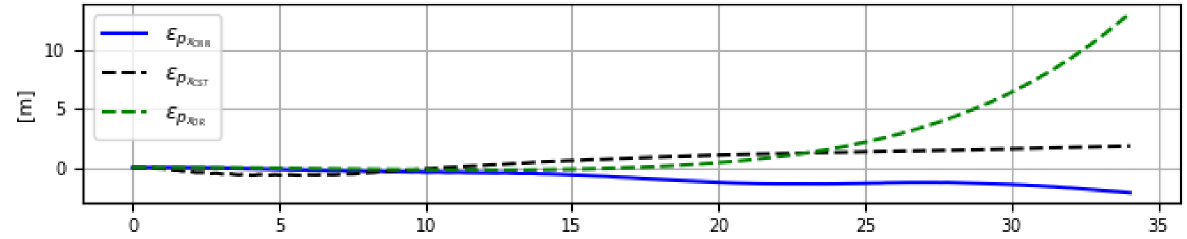
ESTIMATION RESULTS

POSITION ESTIMATION

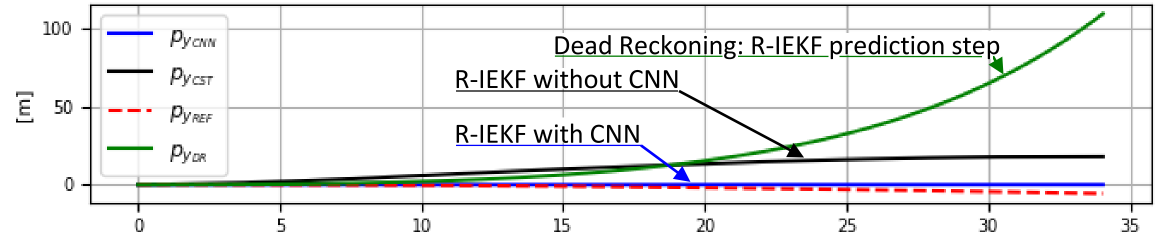
Estimated position along x-axis



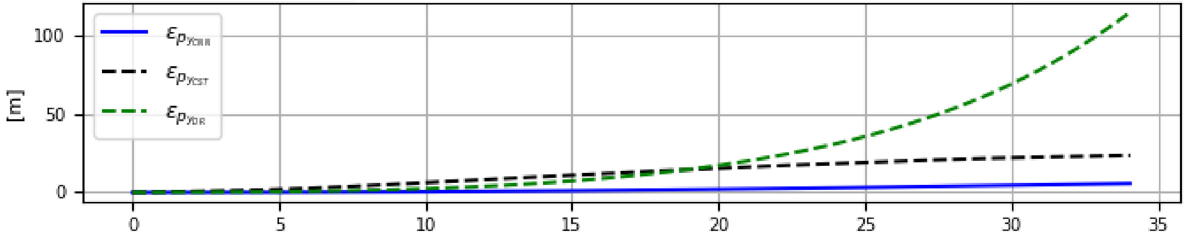
Position errors along x-axis: $\varepsilon_{p_x} = \hat{p}_x - p_{x_{True}}$



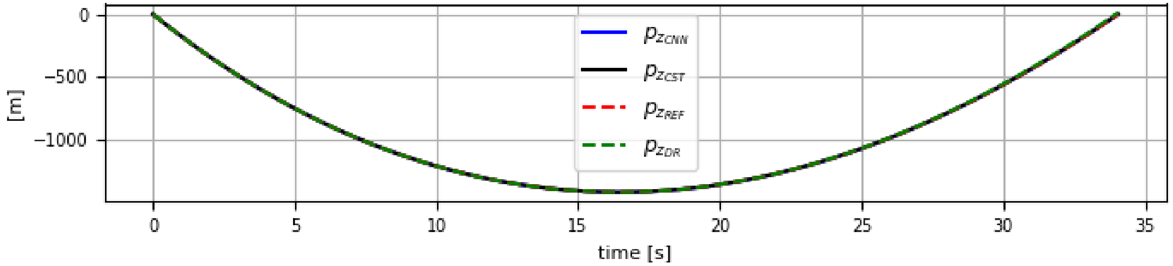
Estimated position along y-axis



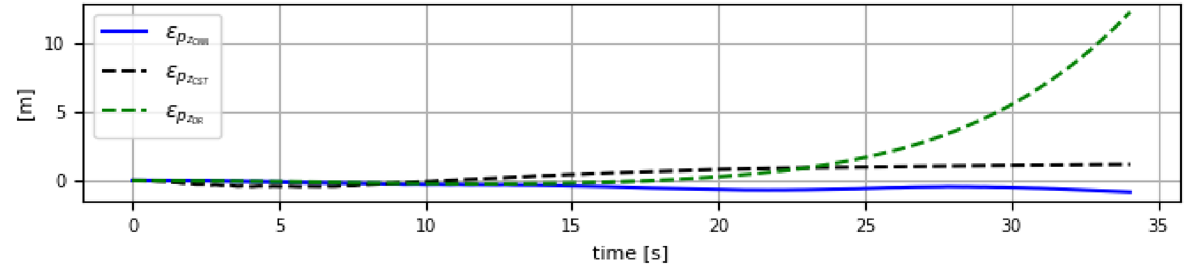
Position errors along y-axis: $\varepsilon_{p_y} = \hat{p}_y - p_{y_{True}}$



Estimated position along z-axis



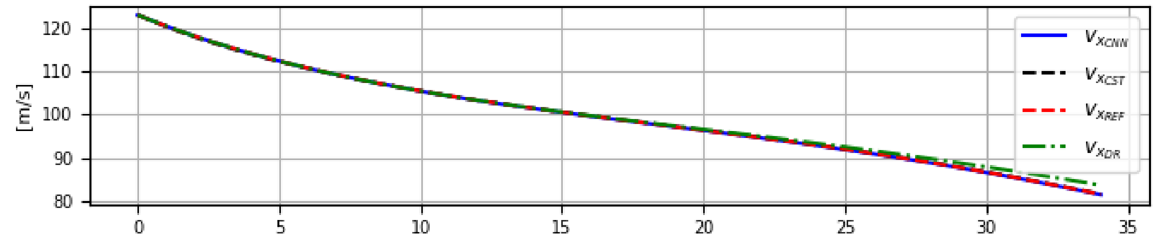
Position errors along z-axis: $\varepsilon_{p_z} = \hat{p}_z - p_{z_{True}}$



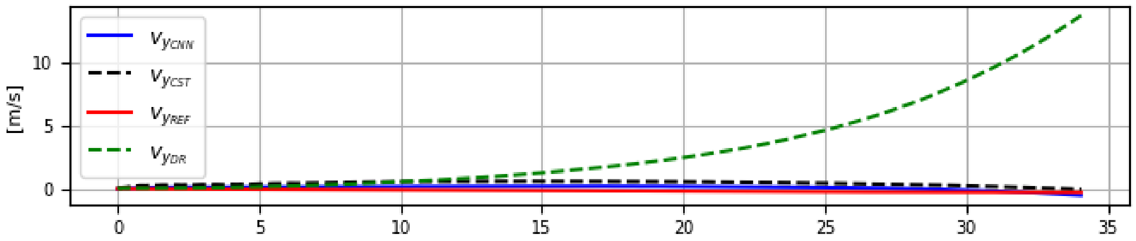
ESTIMATION RESULTS

VELOCITY ESTIMATION

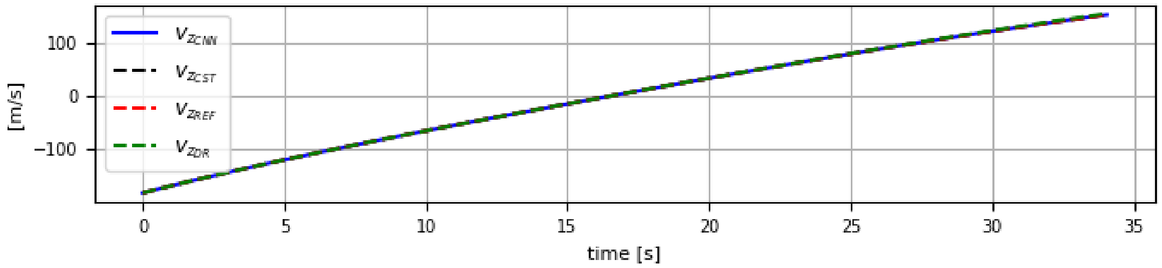
Estimated velocity along x-axis



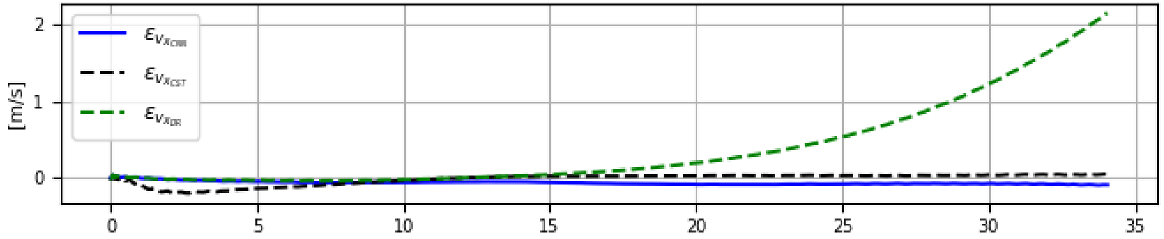
Estimated velocity along y-axis



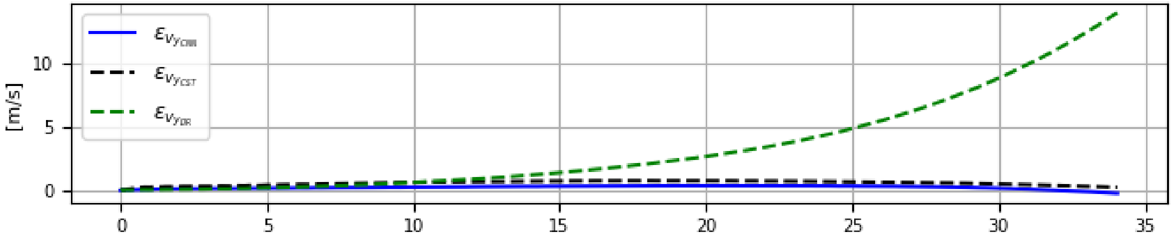
Estimated velocity along z-axis



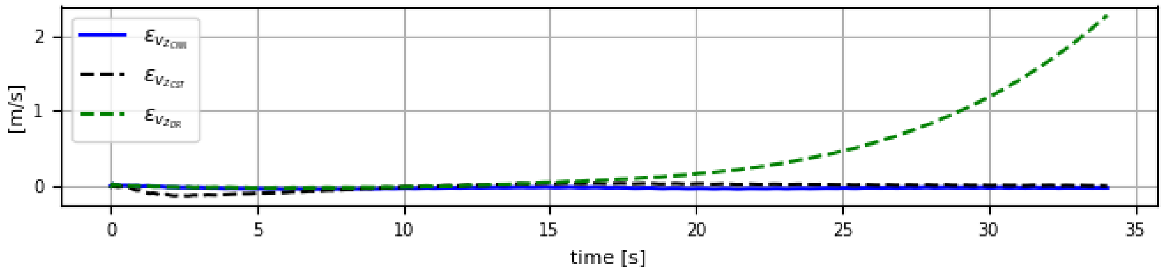
Velocity errors along x-axis: $\varepsilon_{v_x} = \hat{v}_x - v_{x_{True}}$



Velocity errors along y-axis: $\varepsilon_{v_y} = \hat{v}_y - v_{y_{True}}$



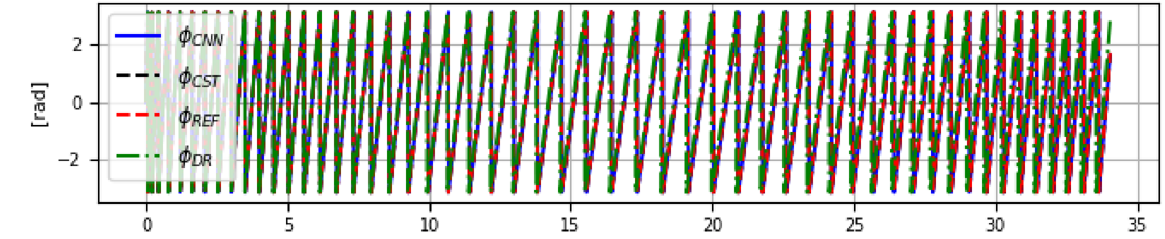
Velocity errors along z-axis: $\varepsilon_{v_z} = \hat{v}_z - v_{z_{True}}$



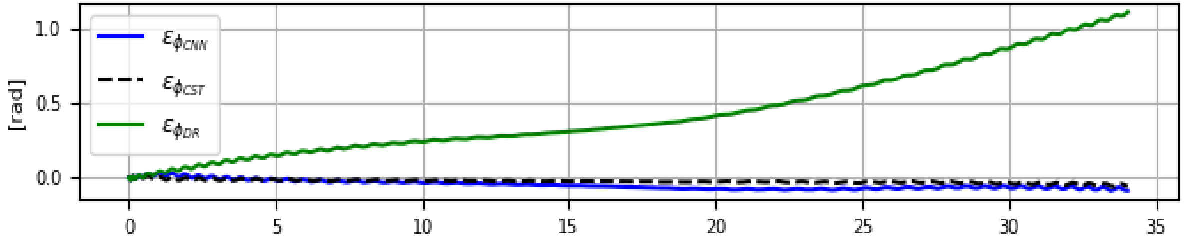
ESTIMATION RESULTS

EULER ANGLES ESTIMATION

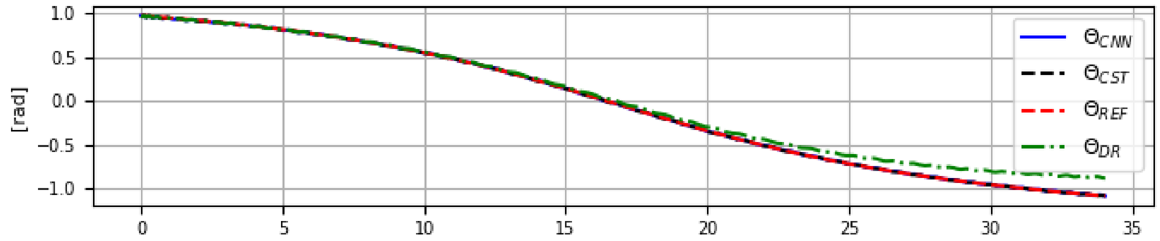
Estimated roll angle



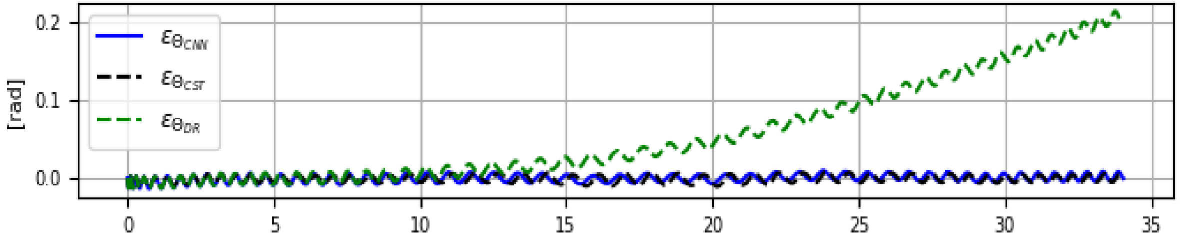
Roll errors: $\varepsilon_\phi = \hat{\phi} - \phi_{True}$



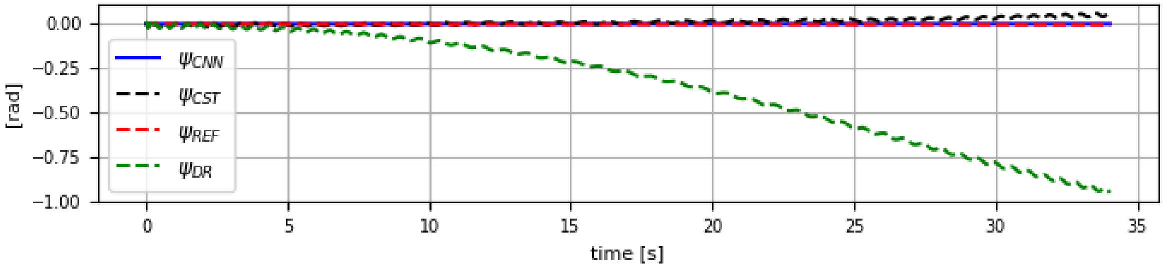
Estimated pitch angle



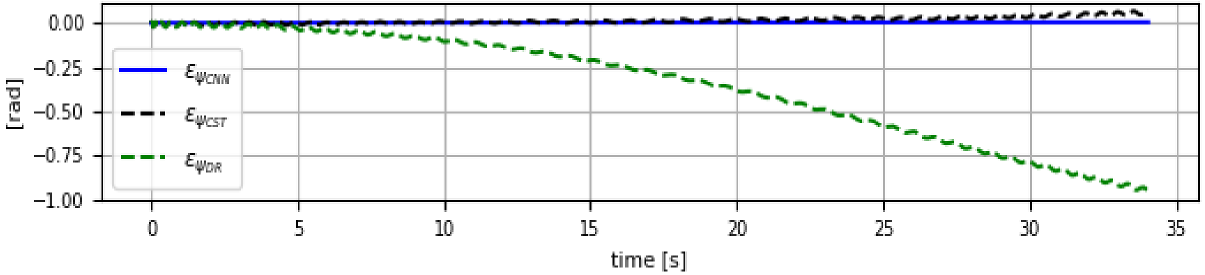
Pitch errors: $\varepsilon_\theta = \hat{\theta} - \theta_{True}$



Estimated yaw angle



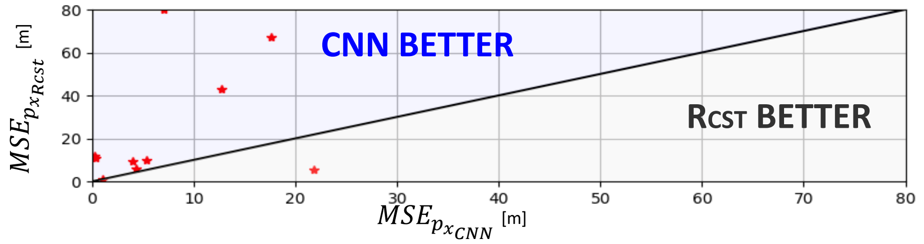
Yaw errors: $\varepsilon_\psi = \hat{\psi} - \psi_{True}$



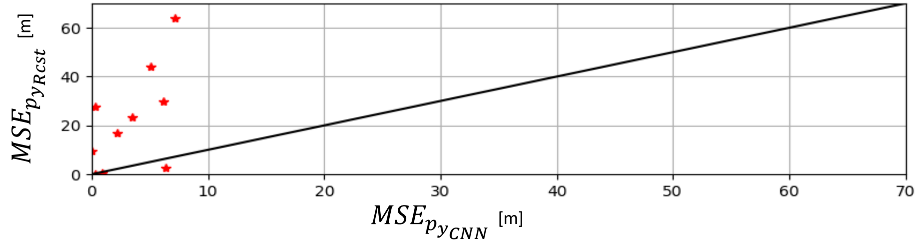
TEST DATASET: ANALYSIS

POSITION

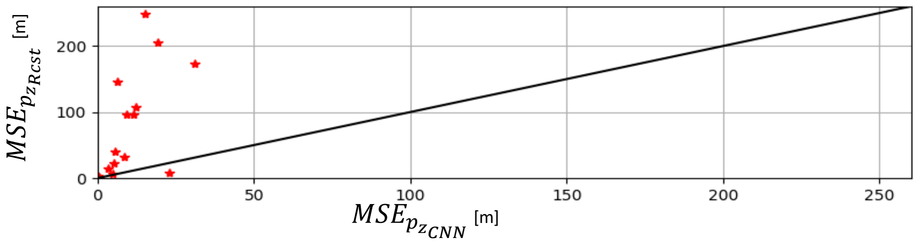
Position errors along x-axis: $\left(MSE_{p_{x_{CNN}}}, MSE_{p_{x_{Rconstant}}}\right)$



Position errors along y-axis: $\left(MSE_{p_{y_{CNN}}}, MSE_{p_{y_{Rconstant}}}\right)$



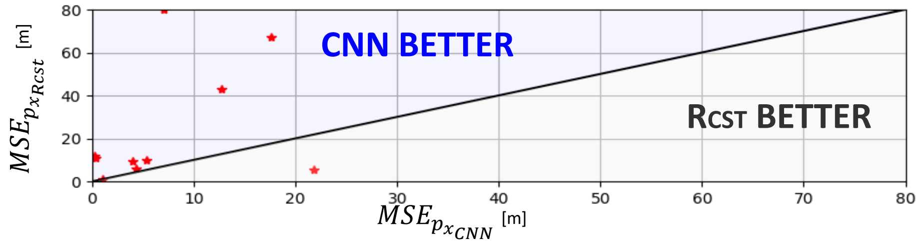
Position errors along z-axis: $\left(MSE_{p_{z_{CNN}}}, MSE_{p_{z_{Rconstant}}}\right)$



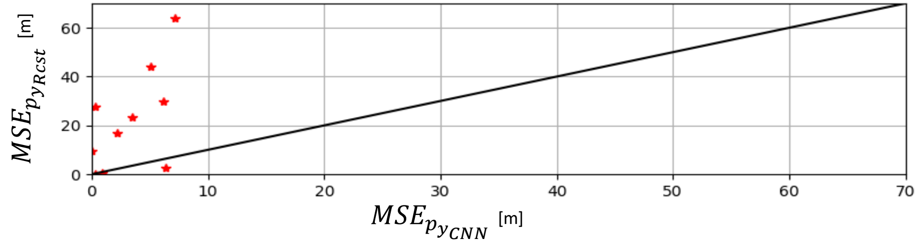
TEST DATASET: ANALYSIS

POSITION

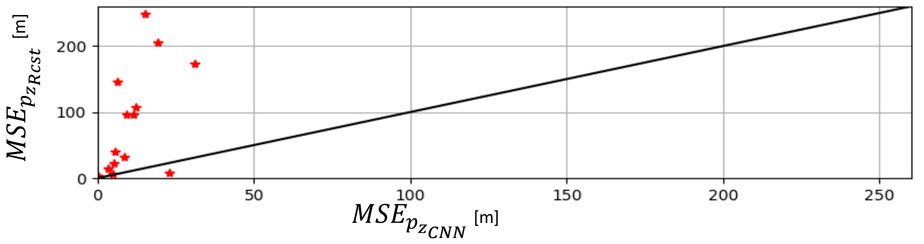
Position errors along x-axis: $\left(MSE_{p_{x_{CNN}}}, MSE_{p_{x_{Rconstant}}}\right)$



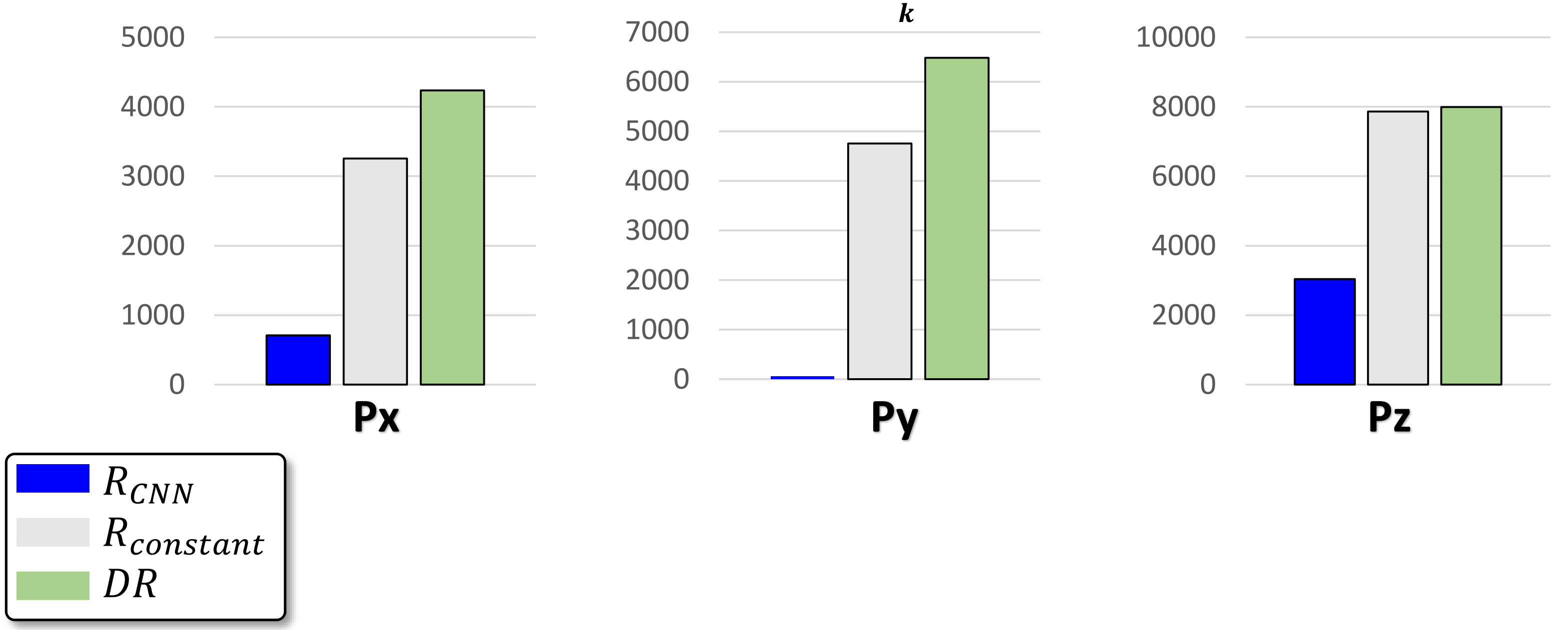
Position errors along y-axis: $\left(MSE_{p_{y_{CNN}}}, MSE_{p_{y_{Rconstant}}}\right)$



Position errors along z-axis: $\left(MSE_{p_{z_{CNN}}}, MSE_{p_{z_{Rconstant}}}\right)$



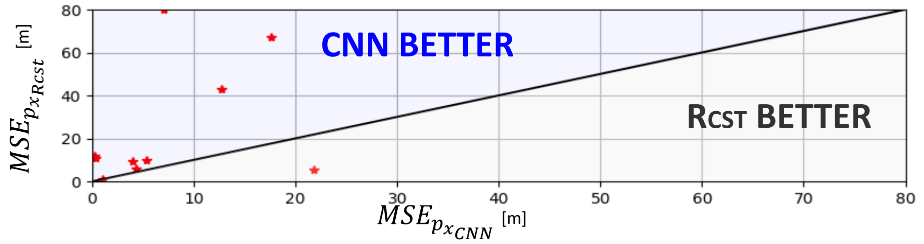
ERROR ANALYSIS $\sum_{k}^{N=20\ simu} MSE_{simu_k}$



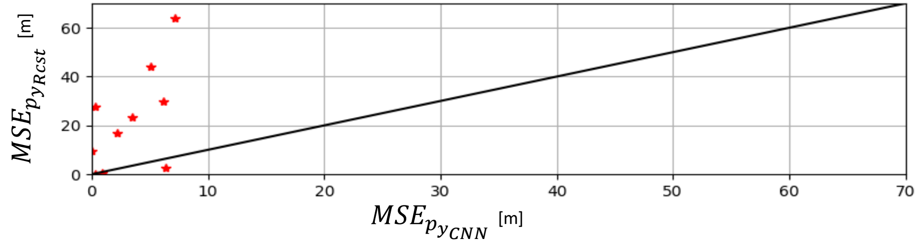
TEST DATASET: ANALYSIS

POSITION

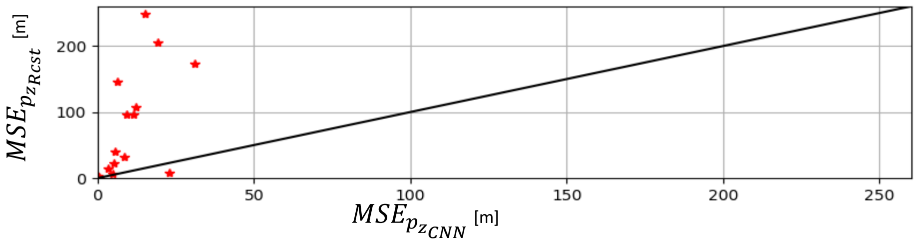
Position errors along x-axis: $\left(MSE_{p_{x_{CNN}}}, MSE_{p_{x_{Rconstant}}}\right)$



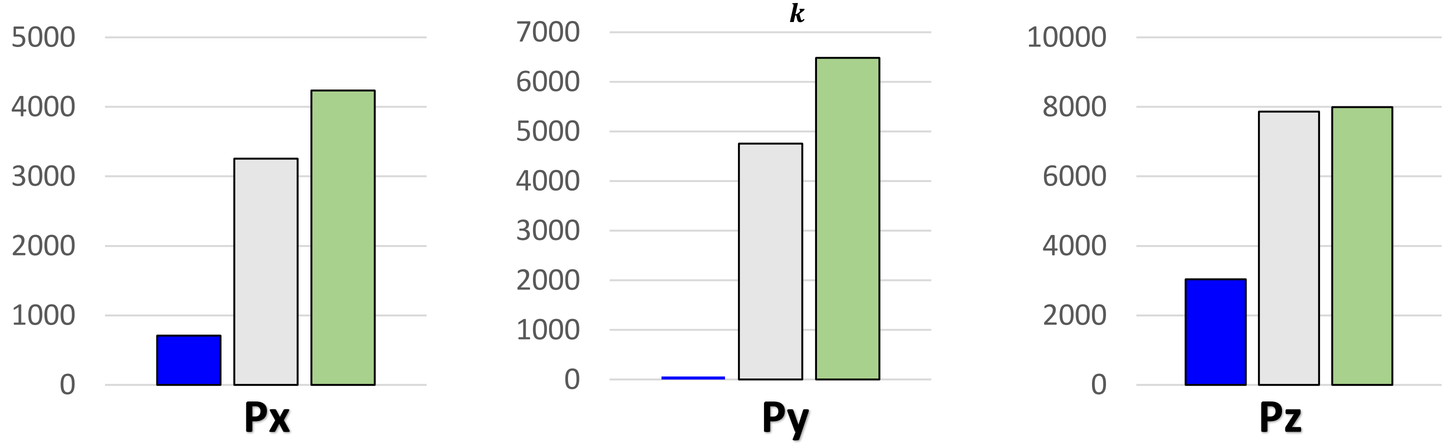
Position errors along y-axis: $\left(MSE_{p_{y_{CNN}}}, MSE_{p_{y_{Rconstant}}}\right)$

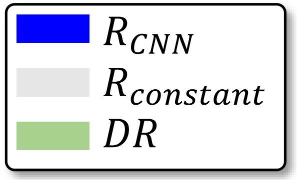


Position errors along z-axis: $\left(MSE_{p_{z_{CNN}}}, MSE_{p_{z_{Rconstant}}}\right)$

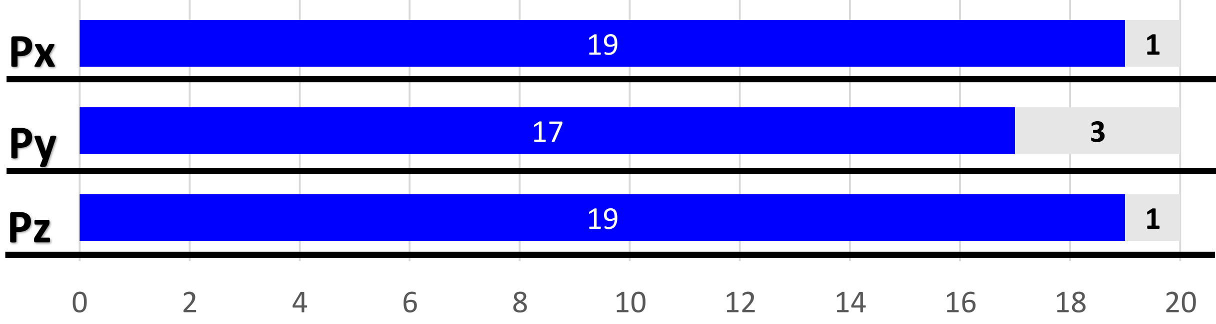


ERROR ANALYSIS $\sum_{k}^{N=20\ simu} MSE_{simu_k}$





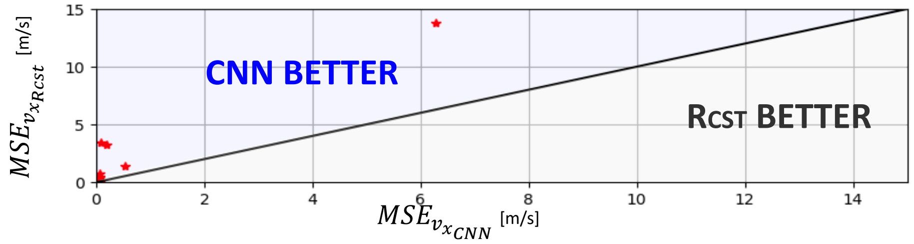
SCORE : $MSE_{CNN} > MSE_{CST}$



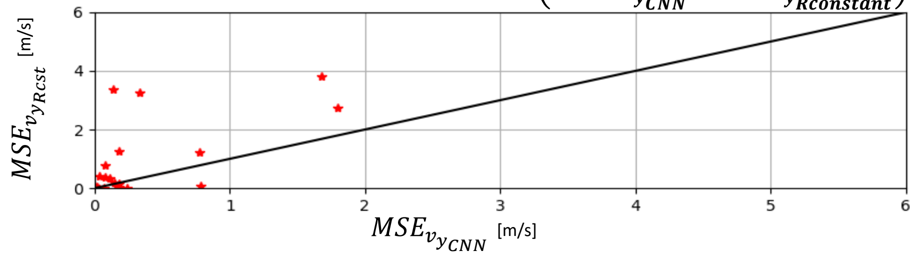
TEST DATASET: ANALYSIS

VELOCITY

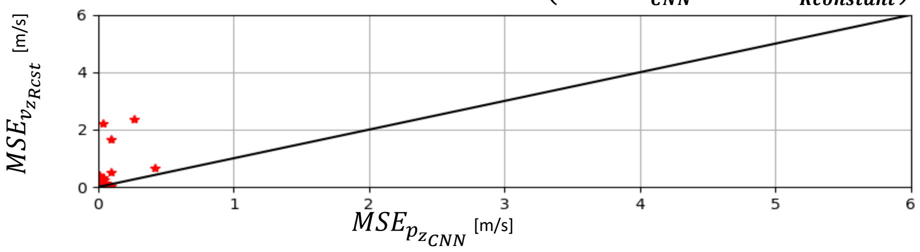
Velocity errors along x-axis: $\left(MSE_{v_{x_{CNN}}}, MSE_{v_{x_{Rconstant}}}\right)$



Velocity errors along y-axis: $\left(MSE_{v_{y_{CNN}}}, MSE_{v_{y_{Rconstant}}}\right)$



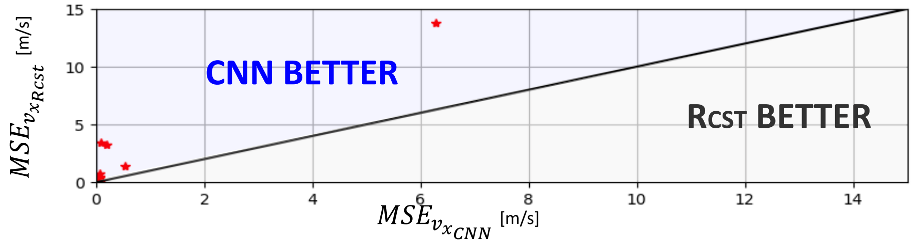
Velocity errors along z-axis: $\left(MSE_{v_{z_{CNN}}}, MSE_{v_{z_{Rconstant}}}\right)$



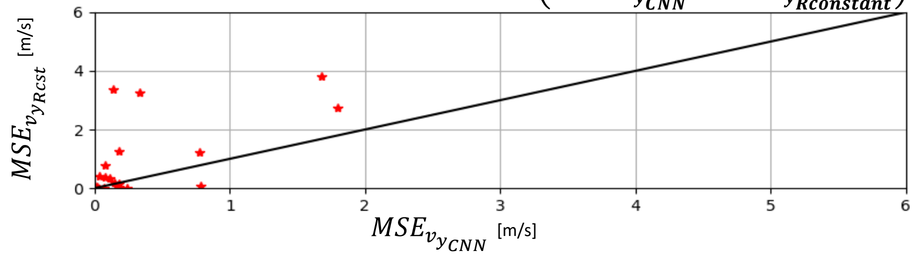
TEST DATASET: ANALYSIS

VELOCITY

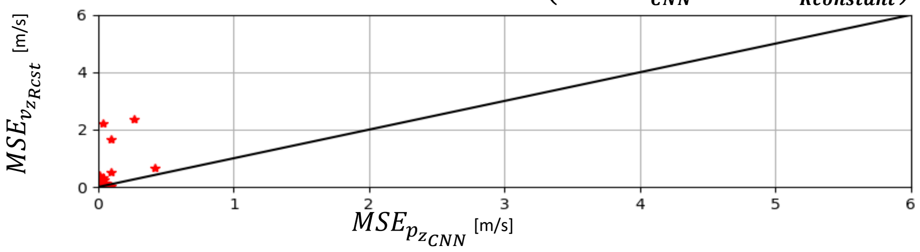
Velocity errors along x-axis: $\left(MSE_{v_{x_{CNN}}}, MSE_{v_{x_{Rconstant}}}\right)$



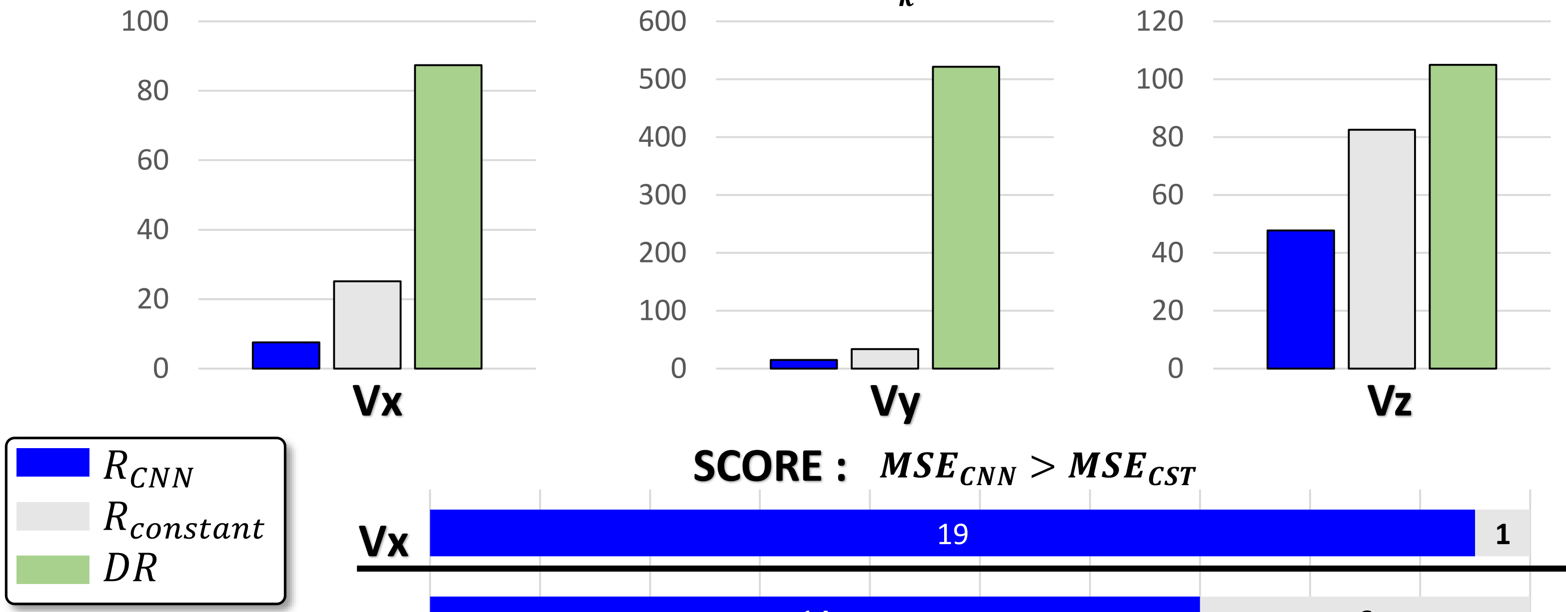
Velocity errors along y-axis: $\left(MSE_{v_{y_{CNN}}}, MSE_{v_{y_{Rconstant}}}\right)$



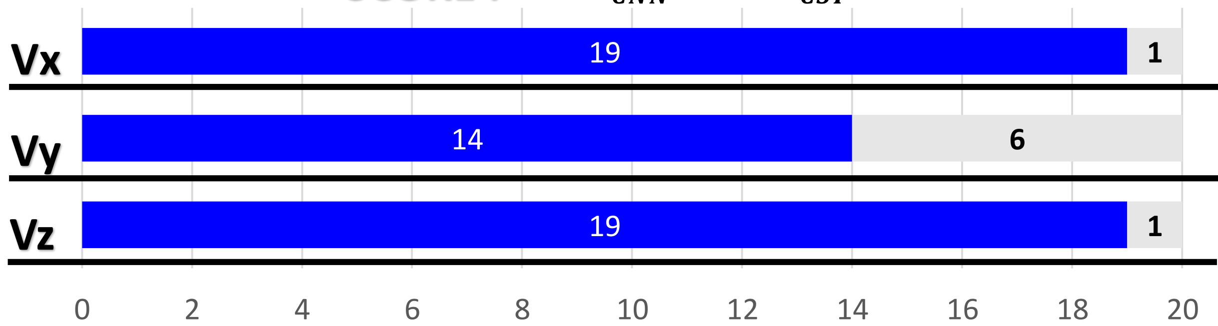
Velocity errors along z-axis: $\left(MSE_{v_{z_{CNN}}}, MSE_{v_{z_{Rconstant}}}\right)$



ERROR ANALYSIS $\sum_{k}^{N=20\ simu} MSE_{simu_k}$



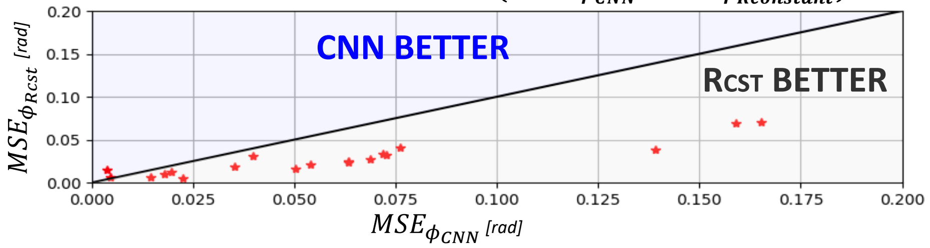
SCORE : $MSE_{CNN} > MSE_{CST}$



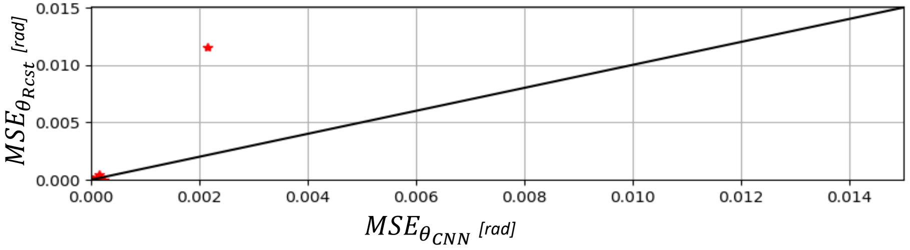
TEST DATASET: ANALYSIS

EULER ANGLES

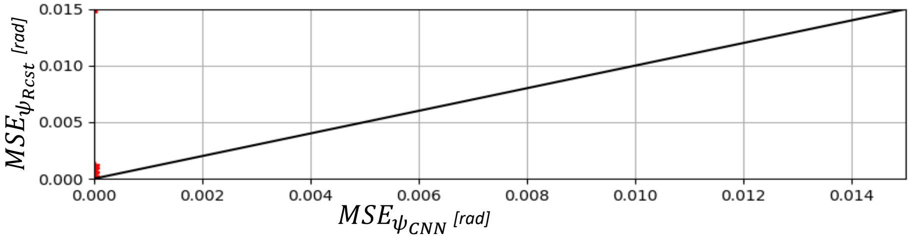
Roll errors along x-axis: $\left(MSE_{\phi_{CNN}}, MSE_{\phi_{Rconstant}}\right)$



Pitch errors along y-axis: $\left(MSE_{\theta_{CNN}}, MSE_{\theta_{Rconstant}}\right)$



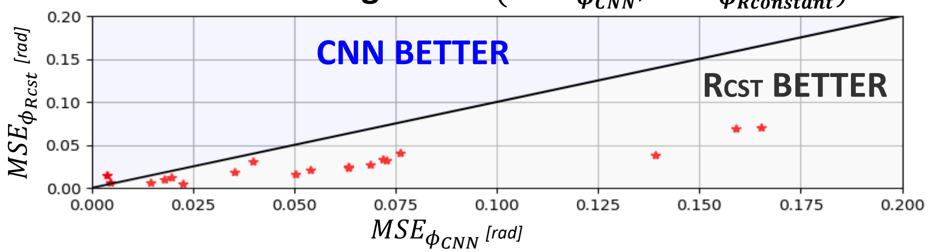
Yaw errors along z-axis: $\left(MSE_{\psi_{CNN}}, MSE_{\psi_{Rconstan}}\right)$



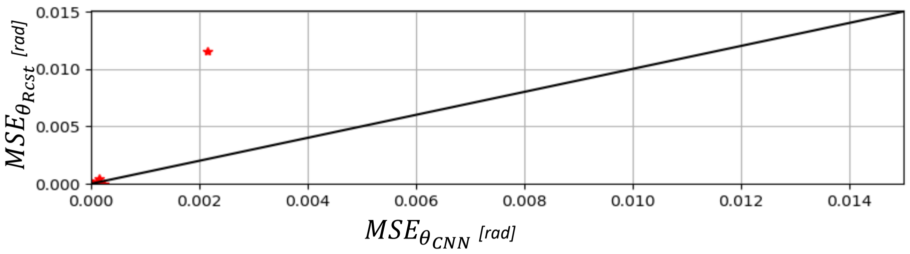
TEST DATASET: ANALYSIS

EULER ANGLES

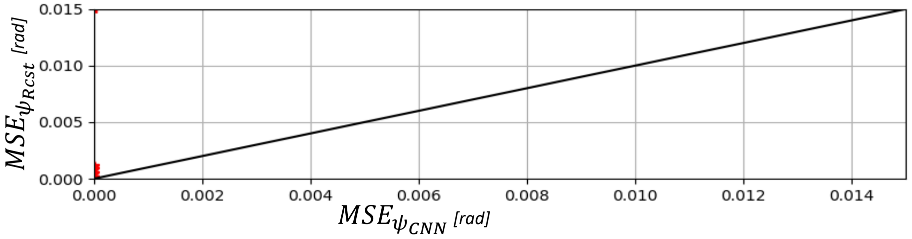
Roll errors along x-axis: $(MSE_{\phi_{CNN}}, MSE_{\phi_{Rconstant}})$



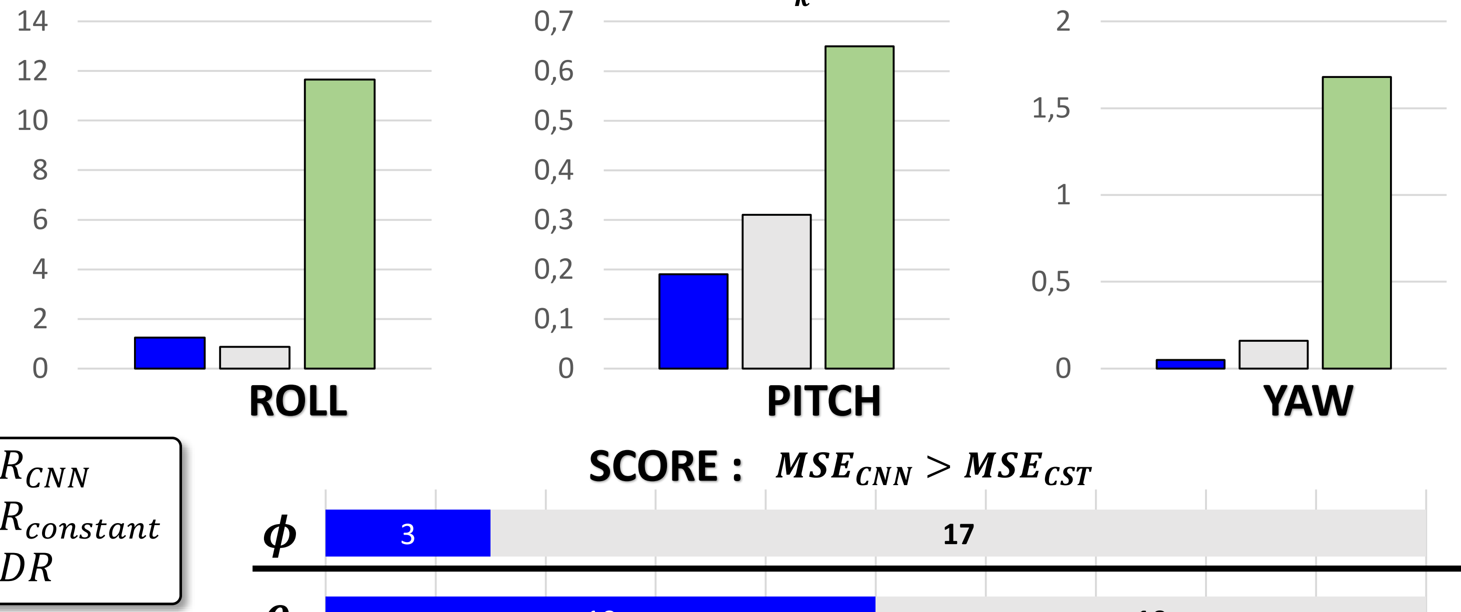
Pitch errors along y-axis: $(MSE_{\theta_{CNN}}, MSE_{\theta_{Rconstant}})$



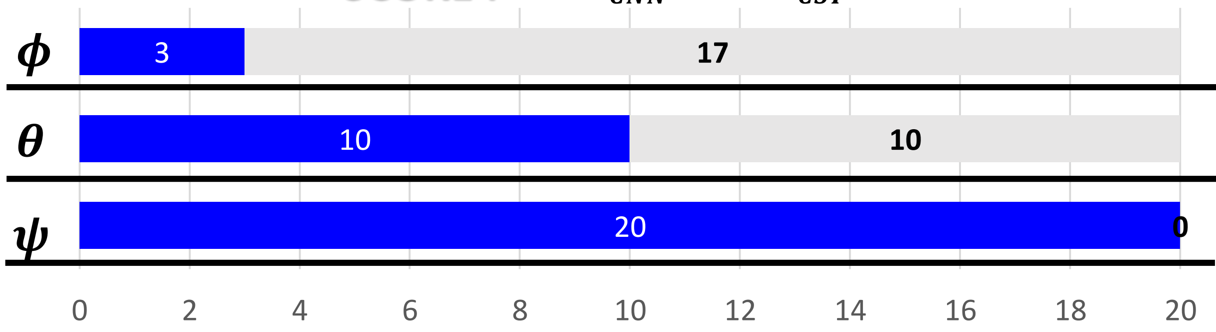
Yaw errors along z-axis: $(MSE_{\psi_{CNN}}, MSE_{\psi_{Rconstan}})$



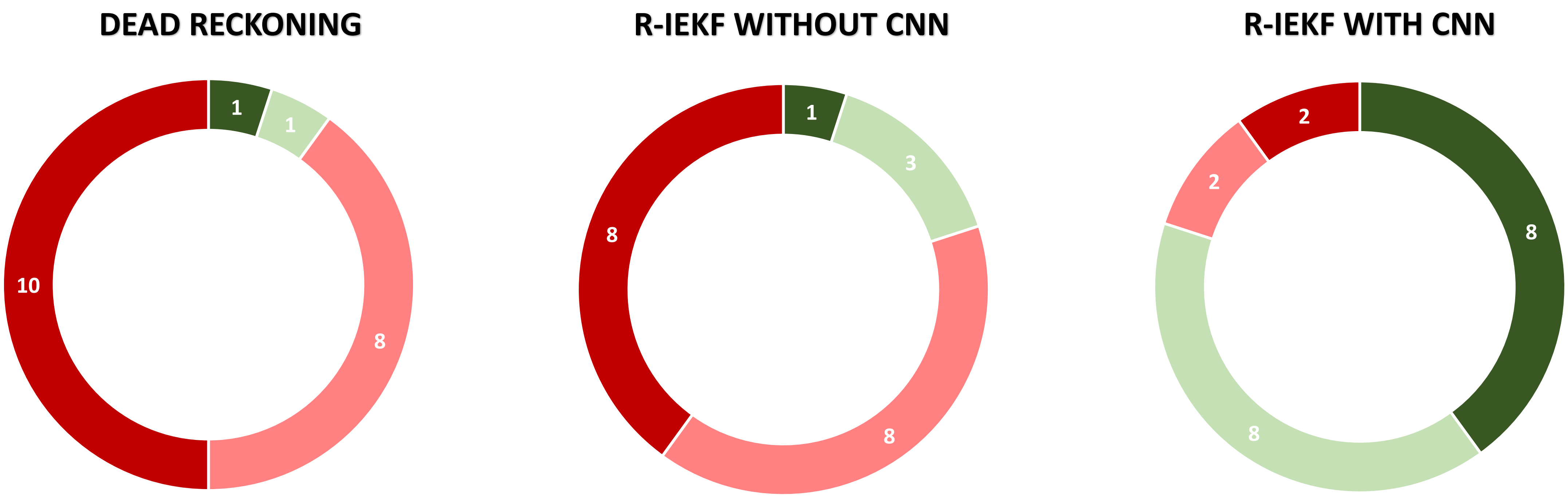
ERROR ANALYSIS $\sum_{k}^{N=20\ simu} MSE_{simu_k}$



SCORE : $MSE_{CNN} > MSE_{CST}$



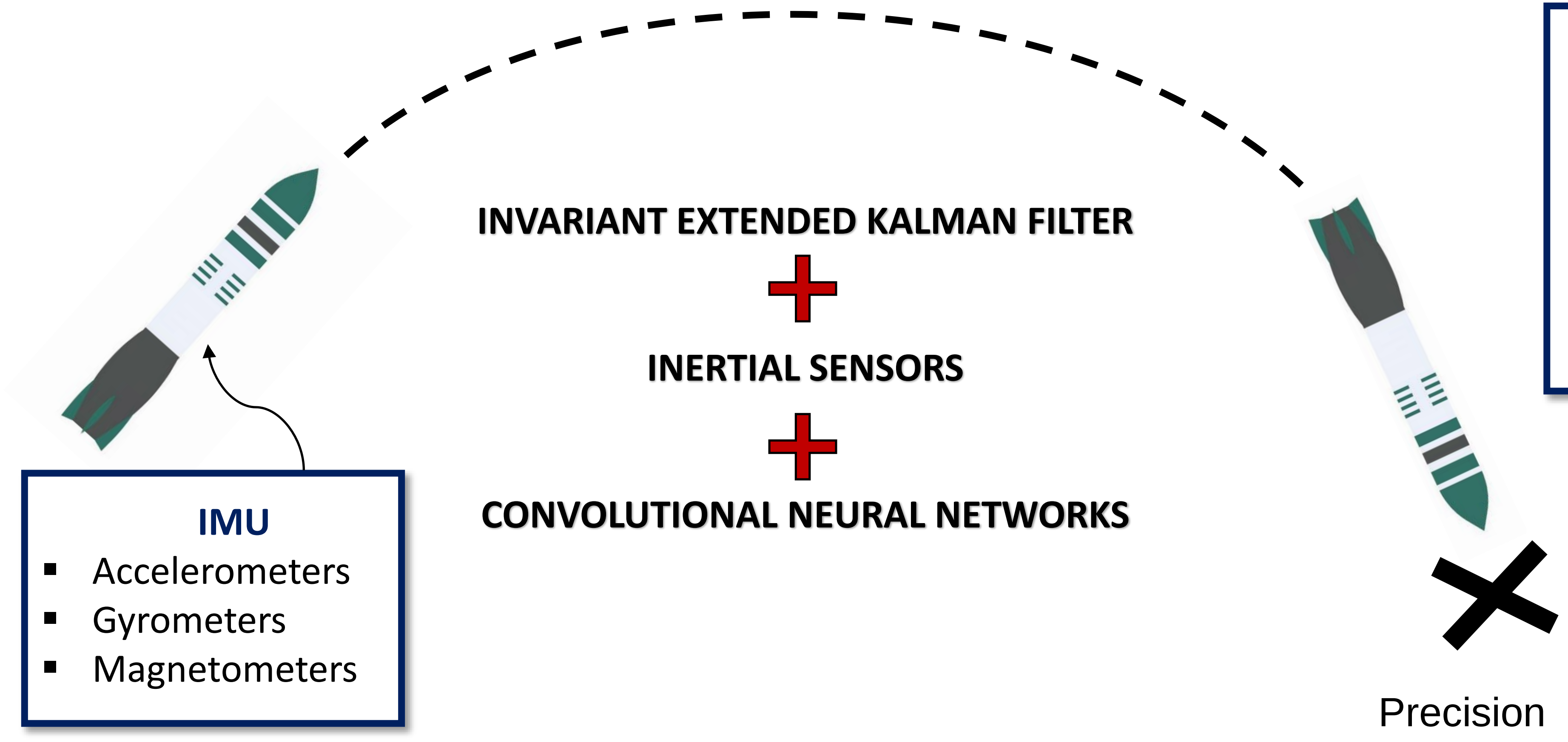
TEST DATASET: ERROR PRECISION



- $(p_{x_{max}}, p_{y_{max}}, p_{z_{max}}) < (5,5,5)\ [m]$
- $(p_{x_{max}}, p_{y_{max}}, p_{z_{max}}) < (10,10,10)\ [m]$
- $(p_{x_{max}}, p_{y_{max}}, p_{z_{max}}) < (20,20,20)\ [m]$
- $(p_{x_{max}}, p_{y_{max}}, p_{z_{max}}) \geq (20,20,20)\ [m]$

V. CONCLUSION

CONCLUSION



IMU
- Accelerometers
- Gyrometers
- Magnetometers

OUTLOOK

- generalization to other projectiles types
- UGV applications

THANK YOU FOR YOUR ATTENTION !